DETECCIÓN DE FRAUDE

Trabajo Fin de Master

Alicia Muñoz Villanueva

ÍNDICE

Introducción

El problema elegido para este Trabajo de Fin de Master es un clásico: **Detección de fraude**. En este caso en concreto, se trata de un problema de detección de fraude de transacciones de dinero. El conjunto de datos ha sido extraído de la plataforma www.kaggle.com.

Se trata de un problema de aprendizaje supervisado. Dentro del aprendizaje supervisado, corresponde a los denominados problemas de clasificación binomial, ya que el objetivo es predecir si una transacción es fraudulenta (1) o no (0). Es decir, la variable objetivo es binaria.

Es uno de los problemas a los que se enfrentan todas las compañías, cuyo objetivo es la detección de fraude para así evitar la pérdida de dinero que ello conlleva. La detección de fraude (junto con la fuga de clientes y detección de enfermedades), es uno de los problemas más típicos de clasificación binomial y que actualmente es algo que se trata en todas las empresas.

El objetivo de este trabajo es un análisis profundo de los datos y realizar diferentes modelos para clasificar transacciones en fraudulentas o no. Se llevarán a cabo pruebas con diferentes modelos y parámetros dentro de cada uno de ellos, para así realizar una comparación, e identificar los mejores resultados. Se utilizarán los estadísticos que mejor se adapten al tipo de problema.

Descripción de los datos de entrada

El conjunto de datos contiene seis millones de registros y once variables, que son:

- **Step:** Unidad de tiempo. En este caso un step, corresponde a una hora. Toma valores enteros de 0 a 743. Con lo que se puede observar que hay 30 días de datos.
- **Type:** Tipo de transacción. Toma cinco posibles valores; *CASH_OUT, PAYMENT, CASH_IN, TRANSFER* y *DEBIT*. Como se puede seguir en el código, tan solo dos (*CASH_OUT* y *TRANSFER*) de los cinco posibles valores tienen presencia de fraude, es decir, que hay tres valores (*PAYMENT*, *CASH_IN* y *DEBIT*) que no están presentes en ningún caso de fraude.
- **Amount:** Importe de dinero de la transacción.
- **NameOrig:** Id de la persona que realiza la transacción.
- **OldbalanceOrg:** Saldo inicial de la persona que realiza la transacción, antes de hacerla.
- **NewbalanceOrig:** Nuevo saldo de la persona que realiza la transacción, después de hacerla.
- **NameDest:** Id de la persona que recibe la transacción. Las personas que empiezan por *M* son "comerciantes".
- **OldbalanceDest:** Saldo inicial de la persona que recibe la transacción, antes de recibirla.
- **NewbalanceDest:** Saldo nuevo de la persona que recibe la transacción, después de recibirla.
- **isFraud:** La transacción es fraudulenta *(1)* o no *(0)*.
- **isFlaggedFraud:** Indica si son transacciones que por términos legales se han detectado a tiempo que son fraudulentas y se han podido parar *(1)* y *(0)* en otro caso.

En el apartado **0.Datos** del código se hace un análisis completo de los datos y variables comentando varias cosas interesantes a partir de los datos obtenidos.

Para más detalle, seguir el código y los comentarios que se encuentran en él.

Metodología

Todos los códigos del trabajo se han realizado en Jupyter y se dividen en dos códigos (notebooks) principales:

0. Datos

Como comentaba en el punto anterior, en esta parte del código se realiza un análisis completo de los datos y variables:

- Se comienza analizando una a una las variables individualmente, tanto gráficamente como analíticamente.
- Después, se busca la relación de cada variable con el target *(isFraud)* y se crean nuevas variables a partir de las existentes.
- Finalmente, se observa la correlación entre variables y se utiliza una técnica de bajo muestreo para cambiar la proporción de fraude de la muestra, ya que inicialmente solo se tiene un 0.12% de fraude.

1. Modelos

- A partir de la base de datos que se obtiene en el punto anterior, se divide la muestra en *Train* y *Test* (con una proporción de 70-30 respectivamente).
- Con ello se implementan varios modelos *(Regresión Logística, Árbol de decisión, Random Forest y Gradient Boosting).*
- Para cada uno de ellos se utilizan un mallado de parámetros, se calculan diferentes formas de evaluación (*Recall, Accuracy, Precision, Auc, …*), dentro de cada modelo, se selecciona aquel con los parámetros que hayan obtenido mejores resultados y se realiza una comparación entre los resultados de los modelos diferentes.
- Debido a que se trata de un problema de fraude, el objetivo es captar el mayor número de fraudes posibles. Por ello se utilizará la medida de *Sensibilidad (Recall)*. Hay una explicación más detallada de esto en la parte correspondiente del código.

En el apartado *"Índice de los códigos"* de esta memoria se puede encontrar un guion de los puntos de cada código.

El trabajo se ha realizado con Python utilizando los Notebooks de Jupyter. Las librerías utilizadas han sido:

- *Pandas*
- *Numpy*
- *Matplotlib*
- *Seaborn*
- *Sklearn*
- *Datetime*
- *Random*

Las funciones que se han utilizado de cada una de las librerías se pueden consultar en el código. Para más detalle, seguir el código y los comentarios que se encuentran en él.

Resumen del resultado

Se han implementado cuatro modelos diferentes, y de estos cuatro modelos se han cogido los que han obtenido mejores resultados respectivamente a partir del mallado de parámetros. Posteriormente, se comparan los diferentes modelos entre sí para ver cuál es el que mejores resultados ha tenido. Cabe destacar que hemos calculado también el tiempo de ejecución para tenerlo en cuenta, pero en este caso no hay gran diferencias entre unos y otros.

Para más detalle ver la parte correspondiente del código, que es 1. Modelos. Dentro del código se encuentra en el apartado 1.7. Conclusiones.

ANEXOS: Índice de los códigos

1. Modelos